How To Build and Deploy LV2 Plugin to MOD Duo

From MOD Wiki

Contents

Introduction

This is a quick start guide to get an LV2 plugin running in a MOD Duo device. Let's cut the chatter and get started.

LV2 Basics

This information is well described elsewhere. The SDK assumes that you have a working lv2 plugin for desktop linux. If you haven't gotten that far, you should start with these links:

- Creating Audio Plugins
- LV2 Book
- Writing Lv2 plugins : An Lv2 Overview

Prepare build tools

In order to run a plugin in the MOD Duo we must compile it for its specific architecture.
The Duo uses an ARMv7 processor running a very basic and stripped-down version of Linux.
Several audio-related libraries are included (fftw, libsndfile, libresample, etc) as well as generic libraries (boost, eigen, qt5core, etc).

We currently provide a custom build system that gives developers a similar system to what's available inside the Duo.

Do not use a regular Linux system, it might lead to issues due to mismatching library versions.

If you're running Linux just clone MOD Plugin Builder and follow the instructions.

In summary:

```
$ git clone git://github.com/moddevices/mod-plugin-builder
$ cd mod-plugin-builder
$ ./bootstrap.sh
```

This process should take at least 1 hour, probably more depending on your CPU. When it finishes you'll be able to build plugins for Duo.

Using docker

Alternatively, if you're familiar with docker or are not running Linux, you can also use our mod-plugin-builder image which includes an already built system.

See this HowTo for more information about docker and mod-plugin-builder.

Build using buildroot '.mk' files

Get source code and create a '.mk' file

In case you haven't started your LV2 plugin yet just please follow through the links above in #LV2 Basics.
We have a few plugin examples available here. For this guide we'll use the eg-amp.lv2 example, already included inside of mod-plugin-builder.

Assuming you have a working LV2 plugin code, you'll now need to create a buildroot mk file to build it.
Buildroot requires you to create the new mk file as `plugins/package/NAME/NAME.mk` - using the same name for both the folder and mk file.

The documentation for this file type is quite extensive, so it's not possible to cover everything here. On the mod-plugin-builder repository (that you should have cloned before) there are many mk file examples under plugins/package/.

TIP: The eg-* "packages" have their plugin code stored locally instead of downloading from external sources.

Compile it

We're all set to compile.

```
$ cd ~/mod-plugin-builder/
$ ./build eg-amp-lv2
$ ls ~/mod-workdir/plugins/
eg-amp.lv2
```

Success, the plugin has been built. If you get an error of no-such file or directory, check that you have set WORKDIR the same as when you bootstrapped the plugin-builder.

Few things about the build script and Buildroot

Buildroot is based on packages to build things. An LV2 plugin becomes a package and because of that it must comply with Buildroot rules.
A few important notes:

- The package name is defined by the folder name and cannot contain '.'
- There must be a <packagename>.mk file inside the package folder
- The package name and '.mk' file name must be the same
- Inside the '.mk' file all defined variables must start with the package name in uppercase replacing '-' with '_'
- You need to define the generated plugin bundle names in the <PACKAGE_NAME>_BUNDLES variable
- Browse through other examples so you get an idea of other variations of the makefiles (how to use cmake or waf for example)
- If you want to rebuild after a change to your plugin or the .mk then it is often easiest to just delete the previous build's directory for your plugin ~/mod-workdir/plugins-dep/build/<packagename>-<version>

Alternatives to retrieve source code

The '.mk' file will define how your source code is retrieved.
In most of the existing packages the '.mk' file tells buildroot to download the source. You can use a local tarball or point directly to the source code if you wish.

Point directly to the source code

Just replace the top section with the following:

```
<PACKAGE_NAME>_SITE_METHOD = local
<PACKAGE_NAME>_SITE = /path/to/source
```

Make sure to remove the <PACKAGE_NAME>_SOURCE line, if it exists.

If you try changing the eg-amp-lv2 example don't forget to remove the trailing path from the make command. It should look like this:

```
<PACKAGE_NAME>_TARGET_MAKE = $(TARGET_MAKE_ENV) $(TARGET_CONFIGURE_OPTS) $(MAKE) -C $(@D)
```

If you place the source code in same folder as the '.mk' file you can set the <PACKAGE_NAME>_SITE like this:

```
<PACKAGE_NAME>_SITE_METHOD = local
<PACKAGE_NAME>_SITE = $($(PKG)_PKGDIR)/
```

You can find a working example of such setup here.

Point to a local tarball

If you make a tar.gz file and put it in the same folder as the '.mk' file, you can replace the top section

with the following:

```
<PACKAGE_NAME>_VERSION = 1.0
<PACKAGE_NAME>_SITE_METHOD = file
<PACKAGE_NAME>_SITE = $($(PKG)_PKGDIR)
<PACKAGE_NAME>_SOURCE = eg-amp-1.0.tar.gz
```

If you want to use an arbitrary path just replace the <PACKAGE_NAME>_SITE variable.

Local development

For local development of plugins using buildroot can be bothersome and confusing.
You can use the cross-compiler and toolchain directly instead of going through buildroot methods.
Note that this expects that your source code build system is cross-compile friendly (ie, no hardcoded compiler and paths and uses pkg-config to find extra libraries).
Also this only works on a real Linux system, without using docker.

The setup is as simple as: (adjust as needed)

```
$ . ~/mod-plugin-builder/local.env
$ make
```

The local.env file will setup your Linux compiler environment variables (such as CC, CXX, CFLAGS, etc) to use mod-plugin-builder files.
If everything goes well, the resulting binaries will be ARMv7, MOD Duo compatible.

Quick example plugin

A quick example plugin is available inside mod-plugin-builder in make -C plugins/package/eg-amp-lv2/source/, which works with this cross-compilation setup.
See https://github.com/moddevices/mod-plugin-builder/tree/master/plugins/package/eg-amp-lv2/source

Building this example plugin is as simple as:

```
$ . ~/mod-plugin-builder/local.env
$ make -C ~/mod-plugin-builder/plugins/package/eg-amp-lv2/source
```

That's it! After this the eg-amp.lv2 bundle is ready to be deployed into a MOD unit.

Deploy it

We can deploy the compiled plugin to the MOD using MOD-SDK or manually using curl (advanced).

If you have mod-sdk installed start it up using the target plugin dir as LV2_PATH, like so:

```
$ export LV2_PATH=~/mod-workdir/plugins/
$ modsdk
```

Then open a browser at localhost:9000, select a plugin from the list and use the "deploy" tab to push

the selected plugin's bundle to the Duo.

For advanced users, you can push a bundle to the mod by running this: (adjust as needed)

```
|$ cd ~/mod-workdir/plugins/
|$ tar cz eg-amp.lv2 | base64 | curl -F 'package=@-' http://192.168.51.1/sdk/install
```

That's it! Your plugin is now inside the MOD!